

A General Stochastic Algorithmic Framework for Minimizing Expensive Black Box Objective Functions Based on Surrogate Models and Sensitivity Analysis

Yilun Wang

School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, 611731, China, yilun.wang@gmail.com

Christine A. Shoemaker

School of Civil and Environmental Engineering and School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853, USA, cas12@cornell.edu

We are focusing on bound constrained global optimization problems, whose objective functions are computationally expensive black-box functions and have multiple local minima. The recently popular Metric Stochastic Response Surface (MSRS) algorithm proposed by Regis and Shoemaker (2007) based on adaptive or sequential learning based on response surfaces is revisited and further extended for better performance in case of higher dimensional problems. Specifically, we propose a new way to generate the candidate points which the next function evaluation point is picked from according to the metric criteria, based on a new definition of distance, and prove the global convergence of the corresponding. Correspondingly, a more adaptive implementation of MSRS, named “SO-SA”, is presented. “SO-SA” is more likely to perturb those most sensitive coordinates when generating the candidate points, instead of perturbing all coordinates simultaneously. Numerical experiments on both synthetic problems and real problems demonstrate the advantages of our new algorithm, compared with many state of the art alternatives.

Key words: global optimization; adaptive learning; radials basis function; response surface model; expensive function; high dimensional problem; sensitivity analysis

1. Introduction

1.1 Motivation and Problem Statement

We are considering the optimization problem whose objective function is a computationally expensive deterministic function, i.e. each of its evaluation takes a substantial amount of computing time. Moreover, the objective function $f(x)$ is often nonconvex and considered as a “black box” without analytic or numerically reliable derivative information, i.e. very limited knowledge of $f(x)$ is available. We take the parameter calibration of complex models as an example, where $f(x)$ is the distance between the measured data and the output of a parameterized complex computer simulation model, which are available for evaluation, but not for analytical inspection. This nonlinear regression procedure is to tune the model parameters in order to make the model match the measured data, and resulted model is used for prediction. However, this kind of nonlinear regression problem brings huge computational challenges when the involved simulators are computationally expensive. In particular,

for relatively high dimensional problems considered in this paper, gradients obtained directly using finite differences are often prohibitively expensive because of too many required evaluations of $f(x)$, and can also be easily contaminated by numerical noise.

Our main objective is therefore to design algorithms that can obtain a reasonably satisfying solution using only a small number of function evaluations because finding the “exact” solution is often intractable. For example, in case of parameter calibration of a computationally expensive simulation model, only a small number of runnings of the simulation model are affordable to find the “best” parameter values to make the model output match the measured data.

For this purpose, surrogates (or called response surface, metamodel) are often been used as an efficient adaptive learning to reflect the information of the underlying black box objective function $f(x)$ (Rios and Sahinidis (2009)). While the use of a surrogate model is well suited for the type of optimization considered here, most of this kind of algorithms are known to become less efficient as the problem dimension increases, partially due to the difficulty to construct a surrogate model $s(x)$ with satisfying global approximation property. Most of existing efforts to deal with high dimensional problems are only trying to reduce a high dimensional problem into low dimensional problems either by decomposition or removing nonsensitive parameters, or other similar ideas, and have a lot of limitations in practice due to their relatively restrict assumptions (Shan and Wang (2010)). In this paper, we will generalize an efficient stochastic surrogate based algorithm framework and present an efficient but simple implementation, for relatively higher dimensional problems. The global convergence is proved and its remarkable practical performance is demonstrated by several widely used synthetic testing problems and one real problems arising in calibration of a groundwater model.

1.2 Related Work

For the computationally expensive function we are focusing on, its actual derivatives are rarely available, although we hope that this circumstance will change as automatic differentiation technology advances. But automatic differentiation does not always produce accurate derivatives and it is not applicable when the complete source code for the objective function is not available. On the other hand, choosing an appropriate step size for approximating derivatives by finite differences is itself a difficult undertaking. The difficulties are compounded by the expense of function evaluation and the fact that we could not afford to try a lot of step sizes as many classical optimization algorithms do. Moreover, finite-difference may be unreliable when the objective function is nonsmooth. The above difficulties discourage the adaption of the algorithms requiring derivative information. Therefore, in the following part, we will mainly review derivative-free global algorithms.

Let $f(x)$ be defined in \mathfrak{D} , which is a compact set of \mathbb{R}^d , and assumed to be a closed hypercube in this paper for simplicity. One kind of well-known methods for global minimization is to run a local optimization algorithm from multiple starting points (i.e., use certain multistart procedure). Examples of multistart approaches include Multi Level Single Linkage (MLSL) (Rinnooy Kan and Timmer (1987)), OQNLP (Ugray et al. (2007)) and etc. The adopted popular derivative-free local optimization methods are mostly various direct search methods (Kolda et al. (2003)), including Pattern Search by Torczon (1997), Mesh Adaptive Direct

Search (MADS) (Audet and Dennis (2006)), and Implicit filtering by Gilmore and Kelley (1995). Notice that although many traditional direct search methods (Dennis et al. (1991)) are local search approaches, they are able to escape from local optima to find better to certain degree if not global optimal solutions (Pardalos and Liu (1998)). Surrogate models can be applied to some of the above to improve their performance in cases of computationally expensive objective function. For example, NOMADm, a MATLAB implementation of the MADS has incorporated several response surfaces including Kriging, radial basis functions, Nadaraya-Watson estimator, and support vector machine (Cortes and Vapnik (1995)). The trust-region methods and the pattern search methods using the approximation models were well studied by Dennis and Torczon (1995). Derivative-free trust-region methods for unconstrained optimization (Conn et al. (1997); Powell (2002); Zaslavski and Powell (2006)) typically rely on local quadratic models of the objective function that are built by interpolation using a subset of previously evaluated points.

As for the global search algorithms, one important kind of methods are deterministic global search algorithms, including Lipschitzian-based partition algorithms, for example, the DIRECT (DIvide a hyper-RECTangle) algorithm (Jones (2001); Jones et al. (1993)), and the Branch-and-Bound algorithms, and Multilevel Coordinate Search by Huyer and Neumaier (1999). However, for high dimensional problems, those methods based on partitioning the solution space have a worst case exponential complexity, and therefore are not suitable for computationally expensive functions, though they might be rigorous and are able to give you a provably global solution. Another approach for finding the global minimum of an objective function with multiple local minima is to use heuristic methods or stochastic methods including simulated annealing, evolutionary algorithms (e.g., genetic algorithms, evolution strategies and evolutionary programming), ant algorithms, scatter search by Glover (1998); Laguna and Marti (2003), Covariance Matrix Adaption Evolution (CMA-ES) by Hansen et al. (2003). However, these algorithms are usually designed for functions that are cheap to evaluate. When $f(x)$ is expensive, the algorithms are expected to maximize the information gained, since a huge number of evaluations of $f(x)$ are often not affordable. In such cases, response surface can be applied in order to further improve their performance. For example, SSKm (Scatter Search with Kriging for Matlab) presented by Egea et al. (2009) is able to improve the performance of scatter search on computationally expensive problems by linking a scatter search method with a Kriging interpolation. Radial basis function interpolation has also been used to accelerate an evolution strategy by Regis and Shoemaker (2004). Meanwhile, among response-surface-free algorithms, there also exist some specifically design for relatively high dimensional computationally expensive functions, such as Dynamically Dimensioned Search (DDS) by Tolson and Shoemaker (2007b), which outperforms many other alternatives especially for cases where the objective function $f(x)$ is not smooth, but rough.

Considering that the surrogate can help reduce the expensive function evaluations of $f(x)$ to find its global optimum, ones have been developing many new surrogate based global optimization algorithms in the past years. This kind of algorithms are usually iterative procedures to keep updating the resulted surrogate surface and the key point is how to pick the next function evaluation point in order to balance the two somehow conflicting purposes, improving the accuracy of the response surface and finding the possible minimum of $f(x)$ based on the current response surface. Jones et al. (1998) developed a kriging-based

global optimization method called EGO where the next iterate is obtained by maximizing an expected improvement function. Aleman et al. (2009) used a variant of the EGO method by Jones et al. (1998) to optimize beam orientation in intensity modulated radiation therapy (IMRT) treatment planning. Villemonteix et al. (2009) also developed a kriging-based method called IAGO that uses minimizer entropy as a criterion for determining new evaluation points. Gutmann (2001) used RBFs to develop a global optimization method where the next iterate is obtained by minimizing a bumpiness function and variants of this method have been developed by Björkman and Holmström (2001), and Holmström (2008).

Recently, a kind of efficient Metric Stochastic Response Surface (MSRS, for short) algorithm was proposed by Regis and Shoemaker (2007). The criterion of selecting the next function evaluation is minimizing a merit function, which is a the weighted sum of the response surface value and the distance with the set of previous function evaluation points. A cycle of weights is adopted to either emphasize more on the response surface value or on the distance criteria. Instead of directly minimizing the merit function, they generate a series of random candidate points and then pick the next function evaluation point with the minimal merit function value. It is a computationally efficient way to choose the next function evaluation point and works well in practice. The candidate points might be generated in different ways and the corresponding algorithms might have significantly different performances. For example, in their paper, the candidate points are generated by perturbing the current best solution via either uniform distribution (Global Metric Stochastic Response Surface Algorithm) or normal distribution (Local Metric Stochastic Response Surface). Both algorithms can achieve global convergence almost surely due to the random generation of the candidate points, and the latter one behaviors better than the former one in most cases, especially when the number of function evaluations of $f(x)$ is small.

1.3 Our Contributions and Paper Organization

In this paper, we are focusing on relatively high dimensional multimodal problems ($d \geq 30$), and the allowed maximal number of function evaluations is typically very small (500, for example), considering the dimension of the problem. Our proposed algorithm is an extension and generalization of the Metric Stochastic Response Surface (MSRS, for short) algorithm by Regis and Shoemaker (2007), which has proved to be more efficient for relatively low dimensional problems than many other existing algorithms. In this paper, we further analyze and extend of the key features of MSRS to make it more suitable for relatively high dimensional problems.

First, we extended the MSRS algorithm, in the aspect of generating the random candidate points from which the next function evaluation point is chosen, and still guarantee the global convergence in the probabilistic sense. In particular, while each candidate point is generated by perturbing the current best solution in every coordinate in the original MSRS, we generalize it by allowing the probability for each coordinate to be perturbed can be smaller than 1. In addition, the probability value for each coordinate is not necessarily the same and we propose to make it dependent on the current learning of the underlying objective function $f(x)$. For example, the local sensitivity information is used to set the probability values of different coordinates in this paper. In particular, the sensitivity analysis is performed on the surrogate surface, because we could not afford to perform the sensitivity analysis on

$f(x)$, due to the very limited number of allowed function evaluations of $f(x)$ and the high dimensionality of it. The more sensitive coordinates have higher probabilities to be chosen to be perturbed when generating candidate points.

Secondly, we further analyzed and modified the criteria for selecting the next function evaluation point from random candidate points, introduced by Regis and Shoemaker (2007). We would like to explain that the combination of this criteria with the candidate points method is quite an efficient and reasonable way for relative high dimensional problems, which can be considered as “constrained random search”.

Thirdly, we perform a comparison of our algorithm with many widely used state of the art algorithms including EGO, SSKm, ESGRBF, LMSRBF, Nomads-DACE, DDS, and DY-CORS, for the relatively high dimensional problems. The above algorithms are all designed for computationally expensive functions. To our best knowledge, this kind comparison of the above algorithms has not been done in literatures before.

In Section 2, we first review MSRS by Regis and Shoemaker (2007) and propose our generalizations and modifications, for relatively high dimensional problems. In Section 3, we proved that global convergence is guaranteed after the generalizations. In Section 4, we propose our new implementation of MSRS named “SO-SA”, which makes use of the local sensitivity information when generating the candidate points. In Section 5, we compared the new algorithm “SO-SA” with several state of the art alternative algorithms for several typical test problems and demonstrate its advantages over them. In Section 6, summary and future work will be presented.

2. Revisiting and Extensions of MSRS

2.1 Review of Algorithmic Framework of MSRS

In the paper by Regis and Shoemaker (2007), a metric stochastic response surface (MSRS, in short) method is introduced, where the next function evaluation is chosen from a sequence of random candidate points according to certain criteria, which is a combination of the response surface value and the distance to the previously evaluated points of $f(x)$. In this paper, we are going to generalize it and before moving forward, we first review the algorithmic framework of the MSRS and briefly explain its steps on by one.

The Algorithmic Framework of MSRS:

Inputs:

- (1) A continuous real-valued function f defined on a compact hypercube $\mathfrak{D} = [\vec{a}, \vec{b}] \subseteq \mathbb{R}^d$.
- (2) A particular response surface model, e.g. radial basis functions or neural networks.
- (3) A set of initial evaluation points $\mathcal{I} = \{x_1, \dots, x_{n_0}\} \subseteq \mathfrak{D}$, e.g. a space-filling experimental design.
- (4) The maximum number of function evaluations allowed denoted by N_{\max} .

Output: The best solution encountered by the algorithm.

Step 1 (Do Costly Function Evaluation) Evaluate the function f at each point in \mathcal{I} . Set $n = n_0$ and set $\mathcal{A}_n = \mathcal{I}$. Let x_n^* be the point in \mathcal{A}_n with the best function value and $f_n^* = f(x_n^*)$.

Step 2 While ($n < N_{\max}$)

Step 2.1 (Fit/Update Response Surface Model) Fit/update the response surface model $s_n(x)$ using the data points $\mathcal{B}_n = \{(x_i, f(x_i)) : i = 1, \dots, n\}$.

Step 2.2 (Randomly Generate Candidate Points) Randomly generate t points $\Omega_n = \{y_{n,1}, \dots, y_{n,t}\}$ in \mathbb{R}^d . For each $j = 1, \dots, t$, if $y_{n,j} \notin \mathfrak{D}$, then replace $y_{n,j}$ by the nearest point in \mathfrak{D} . We refer to the points in Ω_n as *candidate points*.

Step 2.3 (Select the Next Function Evaluation Point) The merit function is considering both response surface value $s_n(x)$ and the distance from the previous evaluated points $\mathcal{B}_n = \{(x_i, f(x_i)) : i = 1, \dots, n\}$. The candidate points corresponding to the smallest merit function value, among the t candidate points in Ω_n , is selected as the next evaluation point x_{n+1} .

Step 2.4 (Do Costly Function Evaluation) Evaluate the function f at x_{n+1} .

Step 2.5 (Update Best Function Value) Update the best function value encountered so far, i.e. if $f(x_{n+1}) < f_n^*$, then $x_{n+1}^* = x_{n+1}$; otherwise $x_{n+1}^* = x_n^*$. $f_{n+1}^* = f(x_{n+1}^*)$.

Step 2.6 (Update Information) $\mathcal{A}_{n+1} := \mathcal{A}_n \cup \{x_{n+1}\}$; $\mathcal{B}_{n+1} := \mathcal{B}_n \cup \{(x_{n+1}, f(x_{n+1}))\}$. Reset $n := n + 1$.

Step 3 (Return the Best Solution Found) Return $x_{N_{\max}}^*$.

Like many other surrogate optimization algorithms, MSRS starts by evaluating the expensive objective function $f(x)$ at an initial set of points usually generated by a space-filling experimental design in Step 1. In this paper, the specific space-filling experimental design adopted is the version by Ye et al. (2000), though other experimental design methods might be also applicable here.

Step 2 is the main body of MSRS and it is an iterative procedure until the computational budget, i.e. the number of the maximal allowed function evaluations of $f(x)$ is reached.

Step 2.1 might use any type of response surfaces such as Radial Basis Function (RBF) interpolation (Powell (1992); Buhmann (2003), kriging Cressie (1991); Sacks et al. (1989)), or neural networks. In all the above cases, the complicated objective function $f(x)$ is expressed as a weighted sum of many simple functions. The predictions of $f(x)$ can then be made based on the adopted surrogate surface. In this paper, we adopts the RBF interpolation and a complete introduction to the RBF interpolation was given in Buhmann (2003).

Step 2.2 and Step 2.3 together are about how to determine the next function evaluation point, or called the design point in some literature. This scheme is the key component for any response surface based optimization algorithm. Different surrogate optimizations adopt different criteria to determine the next function evaluation point, usually in the form

of minimizing a so-called merit function. For example, a native way is to let the merit function be the current surrogate and use its minimum as the next function evaluation point. However, it is highly likely that the chosen next function evaluation point is very close to the current best solution which has already been evaluated, if not “exactly” the same, and therefore, the best solution and the surrogate are hardly improved. In such cases, the algorithm quickly converges to a local (global) minimum of the surrogate, which is often not even local minimum of the true function $f(x)$. Therefore the approximation error of the surrogate also needs to be considered when designing the merit function and therefore most popular existing surrogate optimization algorithms try to make the merit function better balance the improvement of the surrogate quality and the exploitation of the current surrogate. For example, EGO by Jones et al. (1998), which is based on Kriging surrogate, makes use of the explicit approximation error formula of the surrogate to estimate where is least well approximated. The next function evaluation point is selected where the kriging predictor value is small (local search) and the kriging mean squared error is high (global search). Putting some emphasis on searching where the error is high ensures that we improve the approximation accuracy of the surrogate surfaces as well as encourage the global search.

However, there is no explicit error estimation formula for many other the surrogate response surfaces such as the radial basis function interpolations. Thus MSRS proposed to adopt the distance with set of the previously evaluated points to estimate the surrogate approximation accuracy of a given point; roughly, the bigger the distance, the larger the approximation error is. Correspondingly, their merit function was a weighted combination of surrogate values and distance to the previous evaluated points and performed well in all of their test problems.

Instead of directly minimizing the merit function, Regis and Shoemaker (2007) proposed to generate a sequence of random candidate points and pick the one with lowest merit function value as the next function evaluation point, originally due to its computational efficiency. In addition, the great practical performance of the candidate point method has also been demonstrated in Regis and Shoemaker (2007), for relatively low dimensional problems ($d \leq 10$). In this paper, we will further analyze the method of candidate points and generalized its generation method, and present more advantages of the candidate points methods, especially for relatively high dimensional problems.

In Step 2.4, the objective function is evaluated at the selected point.

In Step 2.5, we update the current best solution, if the function value $f(x)$ at the the newly selected point is smaller than the function values $f(x)$ at all the previous function evaluation points.

In Step 2.6, we update the sets related with the available functions valuations of $f(x)$, by adding the newly selected function evaluation point.

The efficiency of MSRS mainly depends on two aspects, one is the definition of the merit function that will be further analyzed in Section 2.2, and the other is quality of the generated candidate points that will be analyzed in section 2.3.

2.2 Revisiting and Modification of the Merit Function

The merit function proposed in MSRS by Regis and Shoemaker (2007) consists of the estimated function value from the current response surface $s_n(x)$ and the minimum distance from previously evaluated points. Specifically, the corresponding merit function is a weighted sum of the response surface value and the distance as follows:

$$u(x) = w_n^S \frac{s(x) - s_{\min}}{s_{\max} - s_{\min}} + w_n^D \frac{d_{\min} - d(x)}{d_{\max} - d_{\min}} \quad (1)$$

where $s(x)$ is the current surrogate model, $d(x) = \min(\text{dist}(x, y_i))$ with y_i being the previous evaluated point ($i = 1, \dots, n$) and $d_{\min} = \min(d(x)), x \in \mathfrak{D}$; $d_{\max} = \max(d(x)), x \in \mathfrak{D}$; $s_{\min} = \min(s(x)), x \in \mathfrak{D}$; and $s_{\max} = \max(s(x)), x \in \mathfrak{D}$. The set of nonnegative weights $\{(w_n^S, w_n^D) : n = n_0, n_0 + 1, \dots\}$ where $w_n^S + w_n^D = 1$ is aiming to control the balance of the surrogate surface value criteria and the distance criteria. Once w_n^S is determined, w_n^D is determined correspondingly since $w_n^D = 1 - w_n^S$. The transition between local search and global search depends on the value of the weight w_n^S (correspondingly w_n^D). A large w_n^S encourages the local refining while a smaller w_n^S more encourages the global exploration.

We would like to revisit the merit function from point of view of adaptive machine learning. The two parts of the merit function correspond to “most informative” data point and selecting “most uncertain” data point, respectively. These two goals are in fact not necessarily always conflicting, i.e. the purpose of selecting the “most uncertain” data point is often to avoid the local trapping of the current response surface and help to find the “most informative” data point (the one with lower objective function value) in the following iterations.

About the setting of w_n^S and w_n^D , the cycled weights consisting of large values and small values is proposed by Regis and Shoemaker (2007) and this kind of cycling of the weights is only a deterministic case. In fact, the cycling of weights means to say that we do not know the size of the approximation error of the current surrogate surface. Therefore, without any other prior information, in this paper we propose to just turn to the randomness, i.e. randomly pick a value between $[0, 1]$. It is a simpler way and better reflect the spirit of the definition of the merit function. Prescribed cycled values such as $[0.3, 0.5, 0.7, 0.95]$ in Regis and Shoemaker (2007) are also only kind of arbitrarily set and has no solid theoretical support. Furthermore, we will also adopt a greedy way in this paper. When we find a significantly better new solution with certain w_n^S , we would like to keep use it until we fail to find a significantly better solution. This strategy often works well in the case of high dimensional problems and very limited number of function evaluations.

Step 2.3 (Select the Next Function Evaluation Point) Use the information from the response surface model $s_n(x)$ and the data points $\mathcal{B}_n = \{(x_i, f(x_i)) : i = 1, \dots, n\}$ to select the evaluation point x_{n+1} from the t random candidate points in Ω_n .

Step 2.3.1 (Estimate the Function Value of Candidate Points) For each $x \in \Omega_n$, compute $s_n(x)$. Also, compute $s_n^{\max} = \max\{s_n(x) : x \in \Omega_n\}$ and $s_n^{\min} = \min\{s_n(x) : x \in \Omega_n\}$.

Step 2.3.2 (Compute the Score Between 0 and 1) For each $x \in \Omega_n$, compute $V_n^S(x) = (s_n(x) - s_n^{\min}) / (s_n^{\max} - s_n^{\min})$ if $s_n^{\max} \neq s_n^{\min}$ and $V_n^S = 1$ otherwise.

Step 2.3.3 (Determine the Minimum Distance from Previously Evaluated Points) For each $x \in \Omega_n$, compute $d_n(x) = \min_{1 \leq i \leq n} \|x - x_i\|_2$. Also, compute $d_n^{\max} = \max\{d_n(x) : x \in \Omega_n\}$ and $d_n^{\min} = \min\{d_n(x) : x \in \Omega_n\}$.

Step 2.3.4 (Compute the Score between 0 and 1 for the Distance Criterion) For each $x \in \Omega_n$, compute $V_n^D = (d_n^{\max} - d_n(x)) / (d_n^{\max} - d_n^{\min})$ if $d_n^{\max} \neq d_n^{\min}$ and $V_n^D(x) = 1$ otherwise.

Step 2.3.5 (Determine the Weights w_n^S and w_n^D) Randomly pick w_n^D in $[0, 1]$ and $w_n^S = 1 - w_n^D$.

Step 2.3.6 (Compute the Weighted Score) For each $x \in \Omega_n$, compute $u_n = w_n^S V_n^S(x) + w_n^D V_n^D(x)$

Step 2.3.7 (Select Next Evaluation Point) Let x_{n+1} be the point in Ω_n that minimizes u_n .

2.3 Further Analysis and Extension of Random Candidate Points Method

Here we would like to further analyze why the next function evaluation point is preferred to be picked from the random candidate points rather than directly minimizing of the merit function $u(x)$ defined as (1), besides the original motivation in Regis and Shoemaker (2007) that direct minimization of the merit function is more computationally costly, because $s(x)$ is non-convex and the computation of s_{\max} and s_{\min} is not straightforward.

First of all, random search methods, as we know, have been shown to have a potential to solve high dimensional problems efficiently in a way that is often not possible for deterministic algorithms in Zabinsky (2003). Specifically, if one is willing to accept a weaker claim of being correct with an estimate that has a high probability of being correct, then a stochastic algorithm might provide such an estimate using much less function evaluations than deterministic algorithms. It is very attractive for our cases. Besides easing the curse of dimensionality, the randomness brought by the candidate points also helps avoid the local traps of the nonconvex $f(x)$ and $s(x)$. However, for relatively high dimensional problems, the existing random search strategies might suffer from slow convergence rate, which is becoming even more unacceptable when the objective function is a computationally expensive function. Therefore, the merit function $u(x)$ can be considered as a guidance for the random search in order for better practical convergence rate and we name our scheme of selecting the next function evaluation point as “guided random search”. Finally, we would like to mention that the probability distribution of generating the perturbations can be beyond the normal distribution and uniform distribution adopted by Regis and Shoemaker (2007).

Secondly, when determine the next function evaluation point, we prefer a more greedy strategy, i.e. searching around the current best solution to obtain an even better solution, due to the very limited total allowed number of function evaluations of $f(x)$ and relatively high dimensionality of the searching space, with the aim to find out a reasonably good enough

solution quickly. In such cases, the method of random candidate points has its advantages over the directly minimizing the merit function $u(x)$, because it can well preserve the already achieved searching progress due to the fact that the candidate points are mostly generated around the current best solution, though the stochastic property still preserves the global exploration, when the normal distribution is adopted to generating the perturbations.

In this paper, we proposed to use a new definition of “neighborhood” in cases of high dimensional problems. In the original implementation of MSRS by Regis and Shoemaker (2007), the neighborhood of the current best solution is measured by the perturbation magnitudes. Here the neighborhood of the current best solution is measured by the number of the coordinates perturbed from the current best solution. That is to say, a point is closer to the current best solution than others if it is generated by perturbing a smaller number of coordinates from the current best solution. In particular, a candidate point can be produced via perturbing the current best solution only in a subset of all the coordinates, instead of all of them, as the original implementation of MSRS did. We set the probability of a coordinate being perturbed as a positive number in $[C_1, 1]$, where C_1 is a very small positive constant close to 0. The original implementations of MSRS can be considered as a special case of our new framework, where the probability of being selected is always 1. Notice that each candidate point may perturb different coordinates and tons of candidate points are generated during each iteration. Therefore a relatively diverse set of search directions in each iteration of the algorithm is still obtained and correspondingly the searching space is almost fully explored, though preferably around the current best solution.

Thirdly, the method of random candidate points bring more flexibility than directly minimizing the merit function $u(x)$. We can adaptively generate the candidate points by take the specific learned property of the underlying function $f(x)$ into consideration. We will show how we take the sensitivity information into the consideration for setting the probability value of a coordinate to be selected when generating candidate points, in Section 4.

3. Proof of Global Convergence

While we generalize the way of generating the candidate points in the above section, the global convergence still be preserved and we will prove it in this section. As we have seen, the generalization is mainly in two aspects, where one is that the probability of a coordinate to be selected to be perturbed when generating a candidate point might be a positive value less than 1, not necessarily equal to 1, and the other is the probability distribution of generating the perturbations can be beyond the normal distribution and uniform distribution adopted by Regis and Shoemaker (2007).

In the original paper of MSRS by Regis and Shoemaker (2007), the authors have already proved that if the random candidate points of each iteration satisfy the following two conditions, the algorithm is guaranteed to converge to the global minimum almost surely provided that the algorithm is allowed to run indefinitely. In the following parts, we will show that the random candidate points produced by our generalized way still satisfy these two conditions.

- Condition [1]: For each $n \geq n_0$, $Y_{n,1}, Y_{n,2}, \dots, Y_{n,t}$ are conditionally independent given the random vectors in \mathcal{E}_{n-1}

- Condition [2]: For any $j = 1, \dots, t, x \in \mathfrak{D}$ and $\delta > 0$, there exists $\nu_j(x, \delta) > 0$ such that

$$P[Y_{n,j} \in B(x, \delta) \cap \mathfrak{D} | \sigma(\mathcal{E}_{n-1})] \geq \nu_j(x, \delta)$$

for all $n \geq n_0$. Here $B(x, \delta)$ is the open ball of radius δ centered at x and $\sigma(\mathcal{E}_{n-1})$ is the σ -field generated by the random vectors in \mathcal{E}_{n-1} .

When generating the candidate points, we have followed the Condition [1]. In the following parts, we will show that the generated candidate points of our generalized way still satisfy Condition [2]. Specifically, let the probability of the i -th coordinate being chosen to be perturbed in n -th iteration is denoted as $P_{n,i}$. While $P_{n,i}$ is always 1 in original MSRS, it can be any positive real number in $(C_1, 1]$, where C_1 is a very small positive constant which can be very close to 0. Let $F_{n,j,i} \geq 0$ be the continuous density function for the coordinate i of the j -th candidate point of the n -th iteration when generating the random perturbation. In this paper, we propose that if the density $F_{n,j,i} \geq 0$ function satisfies certain mild conditions, then Condition [2] always holds.

Proposition 1. *Condition [C2] holds, if there exist constants $C_1 > 0$ and $C_2 > 0$, such that the probability $P_{n,i} > C_1$ and the continuous density function $F_{n,j,i} > C_2$, for every $n \in \mathcal{G}^2$ and $1 \leq i \leq d, j = 1, \dots, t$.*

Proof. Define $\psi_{\mathfrak{D}}(\delta) := \inf_{x \in \mathfrak{D}} \mu(B(x, \delta) \cap \mathfrak{D})$, where μ is the Lebesgue measure on \mathbb{R}^d . Observed that for the compact hypercube \mathfrak{D} , we have $\psi_{\mathfrak{D}}(\delta) > 0$ for any $\delta > 0$.

Fix $1 \leq j \leq t, x \in \mathfrak{D}$ and $\delta > 0$. Since we are considering a compact set \mathfrak{D} it follows that $Y_{n,j}$ has a conditional density given $\sigma(\mathcal{E}_{n-1})$ for each $n \geq n_0$ and this is given by

$$g_{n,j}(y | \sigma(\mathcal{E}_{n-1})) \geq g_{n,j}(n \in \mathcal{G}^2, y | \sigma(\mathcal{E}_{n-1})) \geq C_1 C_2^d > 0$$

for all $y \in \mathfrak{D}$. Hence,

$$\begin{aligned} P[Y_{n,j} \in B(x, \delta) \cap \mathfrak{D} | \sigma(\mathcal{E}_{n-1})] &\geq \int_{B(x, \delta) \cap \mathfrak{D}} g_{n,j}(y | \sigma(\mathcal{E}_{n-1})) dy \\ &\geq C_1 C_2^d \mu(B(x, \delta) \cap \mathfrak{D}) \\ &\geq C_1 C_2^d \psi_{\mathfrak{D}}(\delta) > 0, \end{aligned}$$

for any $n \geq n_0$.

So, Condition [C2] also holds. □

Therefore, according to the results in the original MSRS paper by Regis and Shoemaker (2007), if Condition [1] and Condition [2] hold, the global convergence of MSRS will be obtained in a probabilistic sense, and the theorem is restated as follows.

Theorem 2. *Let f be a function defined on $\mathfrak{D} \subseteq \mathbb{R}^d$ and suppose that x^* is the unique global minimizer of f on \mathfrak{D} in the sense that $f(x^*) = \inf_{x \in \mathfrak{D}} f(x) > -\infty$ and $\inf_{\substack{x \in \mathfrak{D} \\ \|x - x^*\| \geq \eta}} f(x) > f(x^*)$ for all $\eta > 0$. Suppose further that the MSRS method generates the random vectors $\{X_n\}_{n \geq 1}$ and $\{Y_{n,1}, \dots, Y_{n,t}\}_{n \geq n_0}$. Define the sequence of random vectors $\{X_n^*\}_{n \geq 1}$ as follows: $X_1^* = X_1$ and $X_n^* = X_n$ if $f(X_n) < f(X_{n-1}^*)$ while $X_n^* = X_{n-1}^*$ otherwise. Then $X_n^* \rightarrow x^*$ almost surely.*

Proposition 1 indicates that the perturbation of a given coordinate can be generated via a large family of probability distributions, not limited to normal distribution and uniform distribution. In the paper by Regis and Shoemaker (2007), the perturbations generated by the normal distribution or uniform distributions were proved to satisfy Condition [2] when all the coordinate are perturbed simultaneously.

While the global convergence is of theoretical significance, its proof here in fact depends on the probability of all the coordinates being simultaneously selected is still a positive value, though probably very small. That is to say, the probability value of each coordinate to be selected is also a balance between encouraging global exploration and local searching. In our cases, where the problem dimension is high and the available function evaluation functions are very limited, we might prefer small probability values, which squint more toward local searching.

4. Using Sensitivity Information when Generating Candidate Points

As for the specific implementation of Step 2.2 in SO-SA, we adopts the new definition of “neighborhood”, which is measured by the number of the coordinates perturbed from the current best solution. The similar idea has been applied in the DDS (Dynamically Dimensioned Search) algorithm by Tolson and Shoemaker (2007b) and the DYCORS algorithm Regis and Shoemaker (2013). Specifically, the random candidate points are obtained by perturbing only a subset of the coordinates of the current best solution. Moreover, the probability values of perturbing different coordinates are the same for each iteration, and decrease as the algorithm reaches the computational budget.

In this paper, we further propose to set different probability values for different coordinates according to the local sensitivity information of the current best solution estimated on the current response surface. The motivation is that along the more sensitive coordinates, we would like to put more dense candidate points in order for the refinements while putting less on less sensitive ones, i.e. setting smaller number of the perturbed coordinates. In particular, we rank the coordinates from the most sensitive ones to least sensitive ones, and the more sensitive ones are assigned a higher probability value to be chosen and the less sensitive ones are given a much smaller value when generating a candidate point. The detailed description of our adopted particular sensitivity analysis methods is presented in Section 4.1 and we call this specific implementation of MSRS as SO-SA.

The detailed description of Step 2.2 in SO-SA is presented as follows.

Step 2.2 (Generate Random Candidate Points)

Step 2.2.1 (Perform a Local Sensitivity Analysis) A prescribed sensitivity analysis method is performed on the response surface $s(x)$ around the current best solution x_n^* .

Step 2.2.2 (Determine the Probability for Each Coordinate to be Selected as the Perturbed Coordinate) According to local sensitivity analysis of the current

best solution, the more sensitive coordinate has a higher probability value and the less sensitive coordinate has a lower probability. The probability of the i -th coordinate for denoted as $p_{n,i} \in [C_1, 1]$, where C_1 is a prescribed positive number, which might be very small.

Step 2.2.3 (Generate Random Candidates Points based on Local Sensitivity Analysis) Randomly generate t points $\Omega_n = \{y_{n,1}, \dots, y_{n,t}\}$ in \mathbb{R}^d . For each $j = 1, \dots, t$, if $y_{n,j} \notin \mathfrak{D}$, then replace $y_{n,j}$ by the nearest point in \mathfrak{D} . We refer to the points in Ω_n as *candidate points*. For each candidate point $y_{n,j}$, its generating procedure is listed as follows.

Step 2.2.3(1) (Select Coordinates to be Perturbed) For each coordinate i ($i = 1, 2, \dots, d$), generate a random value $\omega_{n,j,i}$ via uniform distribution in $[0, 1]$; Denote coordinates to be perturbed as the set $I_{perturb}^{n,j} \doteq \{i : \omega_{n,j,i} \leq p_{n,i}\}$.

Step 2.2.3(2) (Determine the Standard Deviations for Generation of the Perturbations). For each selected coordinate, the perturbation is generated via normal distribution with mean being 0, and the standard deviation being randomly selected from a series of candidate values ranging from large to small. Therefore, different perturbed coordinates might have different standard deviations.

Step 2.2.3(3) (Generate a Candidate Point) Generate the j th candidate point $y_{n,j}$ by $y_{n,j} = x_n^* + \delta_j^n$ where $\delta_{j,i}^n = 0$ for all $i \notin I_{perturb}^{n,j}$ and $\delta_{j,i}^n$ is a normal random variable with mean 0 and standard deviation settled in Step 2.2.3(2), for all $i \in I_{perturb}^{n,j}$.

Step 2.2.3(4) (Ensure Candidate Point is in Domain) If $y_{n,j} \notin \mathfrak{D}$, replace $y_{n,j}$ by a point in \mathfrak{D} obtained by performing successive reflection of $y_{n,j}$ about the closest point on the boundary of the hypercube \mathfrak{D} .

4.1 Local Sensitivity Analysis on the Response Surfaces

While there have existed a variety of local sensitivity analysis methods, ones can choose appropriate methods according to their own preference. We just use the sensitivity indices based on the existing univariate perturbations and bivariate perturbations as showed in this paper.

Given a function $y(x)$ and a point $\bar{x} \in \mathbb{R}^d$, the first kind of sensitivity index is merely the univariate perturbations, or central finite difference, as follows:

$$SI_i^{1,\Delta} = \left| y(\bar{x}^{(i+,\Delta)}) - y(\bar{x}^{(i-,\Delta)}) \right| \quad (2)$$

where

$$\bar{x}^{(i+,\Delta)} = [\bar{x}_1, \dots, \bar{x}_{i-1}, \bar{x}_i + \Delta, \bar{x}_{i+1}, \dots, \bar{x}_d] \quad (3)$$

and

$$\bar{x}^{(i-,\Delta)} = [\bar{x}_1, \dots, \bar{x}_{i-1}, \bar{x}_i - \Delta, \bar{x}_{i+1}, \dots, \bar{x}_d] \quad (4)$$

The second and the third kinds of sensitivity indices are both based on the univariate perturbations together with the bivariate perturbations. We consider the following matrix L^Δ , which consists of elements $L_{i,j}^\Delta$ that is the largest among all possible perturbations along coordinates i and j in terms of the absolute value. The definition of L^Δ is as follows:

$$L^\Delta = \begin{pmatrix} \ell_{1,1}^\Delta & \ell_{1,2}^\Delta & \cdots & \ell_{1,d}^\Delta \\ \ell_{2,1}^\Delta & \ell_{2,2}^\Delta & \cdots & \ell_{2,d}^\Delta \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{d,1}^\Delta & \ell_{d,2}^\Delta & \cdots & \ell_{d,d}^\Delta \end{pmatrix}$$

where

$$\begin{aligned} \ell_{i,j}^\Delta &= \max(|y(\bar{x}^{(i^+,j^+,\Delta)}) - y(\bar{x})|, |y(\bar{x}^{(i^-,j^+,\Delta)}) - y(\bar{x})|, |y(\bar{x}^{(i^+,j^-,\Delta)}) - y(\bar{x})|, |y(\bar{x}^{(i^-,j^-,\Delta)}) - y(\bar{x})|) \\ \ell_{i,i}^\Delta &= \max(|y(\bar{x}^{(i^+,\Delta)}) - y(\bar{x})|, |y(\bar{x}^{(i^-,\Delta)}) - y(\bar{x})|) \end{aligned}$$

$$\begin{aligned} \bar{x}^{(i^+,j^+,\Delta)} &= [\bar{x}_1, \dots, \bar{x}_{i-1}, \bar{x}_i + \Delta, \bar{x}_{i+1}, \dots, \bar{x}_{j-1}, \bar{x}_j + \Delta, \bar{x}_{j+1}, \dots, \bar{x}_d] \\ \bar{x}^{(i^+,j^-,\Delta)} &= [\bar{x}_1, \dots, \bar{x}_{i-1}, \bar{x}_i + \Delta, \bar{x}_{i+1}, \dots, \bar{x}_{j-1}, \bar{x}_j - \Delta, \bar{x}_{j+1}, \dots, \bar{x}_d] \\ \bar{x}^{(i^-,j^+,\Delta)} &= [\bar{x}_1, \dots, \bar{x}_{i-1}, \bar{x}_i - \Delta, \bar{x}_{i+1}, \dots, \bar{x}_{j-1}, \bar{x}_j + \Delta, \bar{x}_{j+1}, \dots, \bar{x}_d] \\ \bar{x}^{(i^-,j^-,\Delta)} &= [\bar{x}_1, \dots, \bar{x}_{i-1}, \bar{x}_i - \Delta, \bar{x}_{i+1}, \dots, \bar{x}_{j-1}, \bar{x}_j - \Delta, \bar{x}_{j+1}, \dots, \bar{x}_d] \end{aligned}$$

Once L^Δ is obtained, we first run eigenvalue decomposition (EVD) of it, which is typically symmetric. For symmetric matrices, singular value decomposition (SVD) and eigenvalue decomposition (EVD) are almost the identical, expect that the singular values are the absolute value of the eigenvalues, and therefore always non-negative. The eigenvalue vector corresponding to the largest-magnitude eigenvalue value is the direction along which the function $y(x)$ changes most dramatically. Therefore, the absolute value of this eigenvalue vector is chosen as the sensitivity measure $SI^{2,\Delta}$. The probability of coordinate i ($i = 1, \dots, n$) selected to be perturbed can be proportional to the magnitude of $SI_i^{2,\Delta}$. Here, the above idea shares much in common with the EVD or SVD based principle component analysis, where the principle components (eigenvectors) explain most of the data variation. The difference is that we are not just generate the random candidate points exactly along the direction of the principle component. Instead, for each coordinate, the perturbation frequency when generating the random candidate is based on the principle component direction.

In this paper, $y(x)$ is chosen as the current response surface s_n , instead of the original function $f(x)$, due to its cheap evaluation. Meanwhile a bigger step size Δ is allowed, since we are interested in the larger neighborhood of \bar{x} and do not want to be stuck in the small vicinity of \bar{x} .

Now we have two sensitivity measures $SI^{1,\Delta}$, and $SI^{2,\Delta}$. They might not coincide with each other, but they all provide us with some useful ranking information of each coordinate. Therefore, on each iteration $n \geq n_0$, some candidate points are generated following the information of $SI^{1,\Delta}$, some are following the information of $SI^{2,\Delta}$.

5. Computational Experiments

In this section, we will compare with state of the art alternative algorithms designed for minimizing computationally expensive functions, on extensive benchmark test problems and a real problems. A brief overview of these alternative algorithms is first given as follows:

5.1 Alternative Optimization Algorithms

The alternative algorithms to be compared include an original implementation of MSRS by Regis and Shoemaker (2007), named LMSRBF, where all coordinates are perturbed simultaneously via normal distributions when generating a candidate point and the response surface is radial basis function interpolation. We will briefly introduce other alternative algorithms, which come from many different categories of algorithms for global optimization. Most of them have also been incorporating the idea of the response surface approximation for the purpose of minimizing computationally expensive functions, except the dynamically dimensioned search algorithm (DDS, for short) developed by Tolson and Shoemaker (2007b), which is a response surface-free algorithm for computationally expensive objective functions. For the limitation of length, we could not cover all of the related algorithms. For example, the RBF method by Gutmann (2001) and its variants are not included in this comparisons partially because previous work by Regis and Shoemaker (2007) has showed that the LM-SRBF algorithm performed much better than the RBF method by Gutmann (2001) on a wide variety of test problems.

5.1.1 Evolutionary Algorithms

There are three main types of evolutionary algorithms, i.e., genetic algorithms, evolution strategies and evolutionary programming algorithms. In this study, we choose ESGRBF by Regis and Shoemaker (2004) and SSKm by Egea et al. (2009), since they have been tailored for minimizing computationally expensive functions by incorporating the response surfaces to mimic the original function $f(x)$.

ESGRBF belongs to the evolution strategies (ES, for short) and it establishes an RBF model using information from previously evaluated points and uses it to screen out offspring that are promising, thereby reducing the computational effort required in the ES. For a (μ, λ) -Evolution Strategy (or simply (μ, λ) -ES) (Bäck (1995)), each generation consists of μ parents that produce λ offspring via crossover and mutation. In the ESGRBF algorithm, we fit an RBF model to screen out promising offspring. In particular, an (m, μ, λ, ν) -ESGRBF is essentially a (μ, λ) -ES that uses an RBF model to estimate the objective function values of all the offspring, find out the ν offspring with the best RBF approximation values, and perform function evaluation of the expensive objective function only on the selected offspring. The μ parent solutions for the next generation are then selected to be the best μ solutions from the ν offspring that are evaluated in the current generation. The parameter m is the size of the initial experimental design, which is used to initialize the RBF model.

Scatter search (Glover (1998)) is an another important evolutionary approach which originated from strategies for creating composite decision rules and response surface constraints. A new version of scatter search called SSKm (Scatter Search with Kriging for Matlab) by

Egea et al. (2009) presented for computationally expensive objective functions. The Kriging response surface implemented in SSKm helps avoid wasteful evaluations that are unlikely to provide high quality function values, thus effectively reducing the number of true function evaluations required to find the vicinity of the global solution.

5.1.2 MultiStarts+LocalMinimization

Some of the local minimization algorithms performed reasonably well and even better than some algorithms meant for global optimization if the goal is to make quick progress within a relatively limited number of function evaluations. The multi-starting scheme is only performed to determine the starting points for the local minimization runs (i.e., through a space-filling design at the beginning or to select the points for the restarts), though the restarts is performed only very few times, or even not any, due to the very limited number of function evaluations of $f(x)$.

In the numerical experiments below, we use the following local minimization solver, the NOMADm software <http://www.gerad.ca/NOMAD/Abramson/nomadm.html>, which is a Matlab implementation of the Mesh Adaptive Direct Search (MADS) algorithm by Audet and Dennis (2006). In this study, we run NOMADm with the option that uses the DACE response surface model by Lophaven, S.N. and Sondergaard (2002) to make it suitable for computationally expensive functions. DACE stands for “Design and Analysis of Computer Experiments,” which is a methodology that uses kriging interpolation to model the outcome of a computer experiment. When searching over the Kriging response surface, we choose to use the Evolution Strategy with Covariance Matrix Adaptation (“CMA-ES”), which adopts a covariance matrix to explicitly rotate and scale the mutation distribution that works as strategy parameters. This particular implementation of “MADS” was referred as NOMADm-DACE in this paper. The MADS algorithm has incorporated schemes to get out of the local trap to find an even better solution to a certain degree, though it is not necessarily the global solution.

5.1.3 EGO

The efficient global optimization (EGO, for short) by Jones et al. (1998) has been widely used for minimizing computationally expensive functions. The EGO algorithm begins by first generating a number of function evaluations at points generated via a Latin Hypercube (i.e., a space-filling design), and utilizes the DACE stochastic process model, i.e. Kriging to approximate $f(x)$ based on these available function evaluations, where “DACE” is an acronym for “Design and Analysis of Computer Experiments”. EGO consists of two main components. The first one is that given data points, how to establish a Kriging model as follows:

$$r(x) = \mu + \sum_{i=1}^p b_i \exp \left[- \sum_{h=1}^d \theta_h |x_h - x_h^{(i)}|^{p_h} \right] \quad (5)$$

where $\theta_h \geq 0, p_h \in [0, 2], h = 1, \dots, d$. The DACE (5) has up to $2d + 2$ parameters: $\mu, \sigma^2, \theta_1, \dots, \theta_d$ and p_1, \dots, p_d , and b_i can be easily calculated once we have obtained these $2d + 2$ parameters. The second component is related with the determination of the next function evaluation point based on the current Kriging model via maximizing what is called “the

expected improvement”, which aims to weigh up both the predicted value of solutions, and the error in this prediction, in order to automatically balance exploitation and exploration.

5.1.4 DDS

Dynamically Dimensioned Search (DDS) algorithm, which is a response surface-free algorithm, was proposed by Tolson and Shoemaker (2007b) and its main idea is to dynamically and randomly select the perturbed coordinates when generating the next function evaluation point based on the current best solution. DDS searches globally at the start of the search and becomes a more local search as the number of iterations approaches the maximum allowable number of function evaluations. The transition from global to local search is achieved by dynamically and probabilistically reducing the number of perturbed coordinates from the current best solution. The selection of the subset of coordinates for perturbation is completely at random without reference to sensitivity information, and the perturbation is generated via normal distribution. DDS is a greedy type of algorithm since the current solution, also the best solution identified so far, is never updated with a solution that has an inferior value of the objective function. Despite its simplicity, DDS has been shown to be a very effective global optimization algorithm for computationally expensive objective functions in many applications.

5.1.5 DYCORS

DYCORS is a specific implementation of MSRS. The major difference between DYCORS and LMSRBF is that DYCORS is more suitable for large dimensional problems (> 30 dimensions) since it does not perturb all variables of the best point found so far in order to create candidate points, but rather borrows the idea of DDS, i.e. each variable is perturbed with probability $P(n) = p_0 \left[1 - \frac{\log(n-n_0+1)}{\log(N_{\max}-n_0)} \right]$ for all $n_0 \leq n \leq N_{\max}$, and where n_0 is the number of points in the initial experimental design, $p_0 = \min(1, 20/d)$, n is the iteration number, and N_{\max} is the maximum number of allowed evaluations for the optimization. Hence, the probability of perturbation for each variable decreases as the optimization advances (as n grows). It is ensured that at least one variable is perturbed.

5.2 Test Problems

5.2.1 Synthetic Problems

These test problems are not really expensive to evaluate, but are widely used for meaningful comparisons of performance for the different algorithms by pretending that these functions are computationally expensive. This can be done by keeping track of the best function values obtained by the different algorithms as the number of function evaluations increases. The relative performance of algorithms on these test problems are expected to be similar to the relative performance of these algorithms on truly expensive functions that have the same general shape as our test problems. Summary of test problems are in Table 1. While the main motivation of our research is on the relatively high dimensional problems, we still also test our new algorithm SO-SA on several low dimensional problems as well as high dimensional problems. We will see that SO-SA is also effective for low dimensional problems.

Test Funs	Domain	Global Min	Test Funs	Domain	Global Min
Ackely30	$[-15, 20]^{30}$	-20-e	Michalewicz30	$[0, \pi]^{30}$	< -23
Rastrigin30	$[4, 5]^{30}$	-30	Schoen35	$[0, 1]^{30}$	< -80
Levy30	$[-5, +5]^{30}$	< -11	TB32	$[0, 1]^{32}$	0
Keane30	$[1, 10]^{30}$	< -0.39			

Table 1: Summary of test problems

5.2.2 Town Brook Watershed Problem

The Town Brook watershed is a 37 km² subregion of the Cannonsville watershed (1200 km²) in the Catskill Region of New York State. The time series Y of measured stream flows and total dissolved phosphorus (TDP) concentrations used in the analysis contains 1096 daily observations (from October 1997 to September 2000) based on readings by the U.S. Geological Survey for water entering the West Branch of the Delaware River from the Town Brook watershed. We used the SWAT2005 simulator (ARNOLD et al. (2012)), which has been used by over a thousand agencies and academic institutions worldwide for the analysis of water flow and nutrient transport in watersheds. The water draining the Town Brook and rest of the Cannonsville watershed collects in the Cannonsville Reservoir, from which it is piped hundreds of miles to New York City for drinking water. Water quality is threatened by phosphorus pollution and, if not protected, could result in the need for a New York City water filtration plant estimated to cost over 8 billion. For this economic reason as well as for general environmental concerns, there is great interest in quantifying the parameter uncertainty for this model. The input information of the Town Brook simulator is discussed briefly in Tolson and Shoemaker (2007b) and in more details in Tolson and Shoemaker (2007a). From the computational point view, in total 32 parameters normalized to $[0, 1]$ will be calibrated and the objective function is the sum of 4 terms which measure the relative distance between the model outputs and the measure data, corresponding to flow, dissolved phosphorus, sediment, and organic nitrogen transported with water, respectively.

5.3 Experimental Setup

5.3.1 Parameter Settings of Involved Algorithms

We perform all numerical computations in Matlab (R2011a) on a Dell workstation with 2.66Ghz CPU and the 32 GB of RAM. The operating system is 64-bit Windows sever 2008 R2 standard. We compared SO-SA with 6 alternatives: LMSRBF, DDS, EGO, SSKm, NOMADm-DACE, ESGRBF.

For LMSRBF, EGO, and ESGRBF, SO-SA, a Latin hypercube design is used to initialize the RBF model with size $m = 2(d + 1)$. For LMSRBF, ESGRBF, SO-SA, we all use a cubic RBF model to approximate the expensive objective function in every iteration.

As for the the ESGRBF algorithm, we set the number of parents $\mu = 8$; the number of offspring $\lambda = 50$ and $\nu = 20$ offspring is chosen according to the RBF model.

SSKm contains several steps. In the improvement step, a local search is implemented using a carefully selected starting point and there are many options for the local search methods such as fmincon in the Matlab optimization toolbox, fminsearch, NOMADm, solnp

(a SQP method by Ye (1987)). In this paper, we are using fmincon. In addition, it contains a kriging interpolation. Its parameter parameter is by default performed by fminsearch, whose the maximum number of function evaluations is by default 200*d. For the rest settings, we also follow its default settings.

As for EGO, we adopted the implementation in the SURROGATES Toolbox developed by Viana (2010). It used the DACE toolbox <http://www2.imm.dtu.dk/~hbn/dace/> to establish the Kriging model in each iteration. When maximizing the “the expected improvement” to determine the next function evaluation point, it used a differential evolution algorithm by Price et al. (2005). The default settings are used in the following experiments.

As for “NOMADm-DACE”, when choosing the search strategy, we select the option of makes use of surrogates, since we are considering the computationally expensive functions. Among the available surrogates, we adopt the DACE, which aims to establish a Kriging surrogate. When searching over the surrogate, we choose to use “CMA-ES”, since it supports global search. The corresponding parameters are kept as defaults.

Since we are considering the computationally expensive functions, we are mainly caring about the number of true function evaluations. In fact, besides these expensive function evaluations, the overhead computation of SO-SA is still much less than EGO, KKim, NOMADm-DACE-CMAES and ESGRBF, though slightly higher than DDS, LMSRBF, as showed in the section 5.5.

5.3.2 Evaluation Criteria

There are never likely to be fully accepted automated stopping criteria for stochastic search algorithms. For that reason, we will generally emphasize algorithm comparisons and stopping based on “budgets” of function valuations in this paper. Given the number of function evaluations which is usually not big due to the high computationally cost of the function evaluation, we compare the lowest objective function values that different algorithms can achieve.

5.4 Results on Test Problems

The results of the proposed SO-SA was compared with these alternatives are plotted in Figures 1, 2, 3, 4, 5, 6, and 7. For each figure, the horizontal axis represents the number of performed function evaluations of $f(x)$ and the vertical axis is the current minimal function value f_n^* among already performed function evaluations. For each problem, 30 runs are performed for each algorithm since they are mostly stochastic algorithms. The averaged current lowest function values and the corresponding standard deviations are plotted. We can see that for all these test problems, our algorithm SO-SA always achieves lowest objective function value, and achieve significant improvements over alternatives mostly. The corresponding size of standard deviation is comparable with these alternatives, if not smaller.

We further proposed a quantify $Q(A, P)$ to better describe the relative performances of different algorithms, where $Q(A, P)$ =the relative difference between the objective function value by algorithm A on problem P and the best objective function value among all algorithms on problem P . That is to say, $Q(A, P) = \frac{|f_{best}^{A,P} - f_{best}^P|}{|f_{best}^P|}$ where $f_{best}^P = \min_A f_{best}^{A,P}$. Let $Q(A) = \sum_P Q(A, P)$ and the smaller $Q(A)$ indicates the better performance of the algorithm

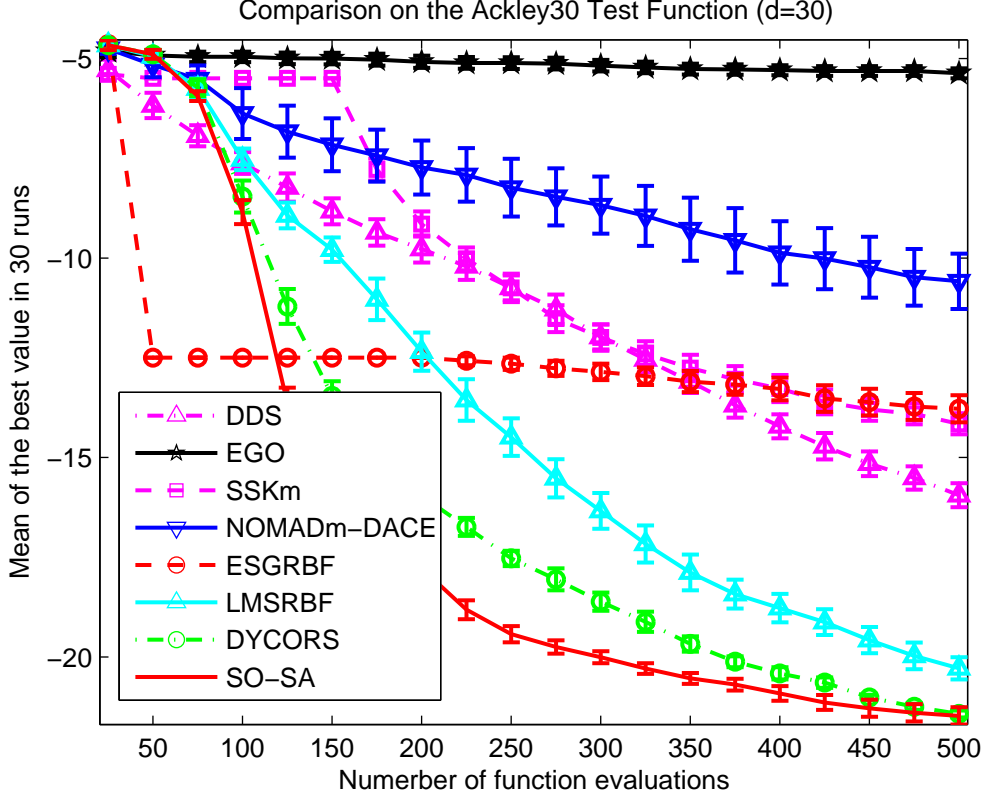


Figure 1: Comparison of Global Optimization Methods on the Ackley Function ($d = 30$)

A. Table 2 summarized the Q values for different algorithms given these testing problems. We can see that SO-SA performs the best compared with these alternatives. We would like to point out EGO performs worse in our cases of high dimensional problems.

5.5 Short Note about the Overhead Running Time

When minimizing the computationally expensive objective function, we usually assume that the running time is mostly spent on the evaluations of the objective function $f(x)$. Specifically, the overhead time including establishing the response surface and determining the next function evaluation is considered to be ignorable compared with the evaluations of $f(x)$. That is to say, this kind of overhead time is mainly for the algorithm to “think” where to perform the evaluations of $f(x)$. However, this is not always true, and the “thinking” time for different algorithms are quite different and even big and Table 3 is the overhead CPU time of our tested algorithms for one run where 500 function evaluations are performed. The “thinking” time of our algorithm is moderate although longer than LMSRBF and DYCORS as expected due to the more complicated schemes for determine the next function evaluation point. Notice that while the “thinking” time of SSKm and EGO is significantly longer than others, their results are among the worst from Table 2, especially for EGO. The overhead time is independent of the running time of each evaluation $f(x)$ and therefore whether it is ignorable often relies on the evaluation time of $f(x)$.

Table 2: Summary of Running Results

	SO-SA	DYCORS	LMSRBF	DDS	SSKm	ESGRBF	NOMADm	EGO
Ackley30	-21.55	-21.41	-20.29	-15.94	-14.15	-13.78	-10.68	-5.37
Rastrigin30	-26.48	-20.88	-16.94	-7.69	4.07	8.16	28.83	113.32
Michalewicz30	-21.30	-19.23	-10.14	-19.56	-11.02	-9.40	-7.22	-7.39
Keane30	-0.40	-0.35	-0.24	-0.30	-0.30	-0.21	-0.21	-0.14
Levy30	-10.45	-9.75	-7.11	-1.14	10.76	11.75	50.32	200.01
Schoen35	-84.39	-82.89	-74.61	-57.72	-44.49	-29.11	-16.26	-8.58
TB32	2.35	2.43	2.47	2.46	2.71	2.53	2.66	2.79
Q(A)	0	0.56	1.83	2.56	4.88	5.56	10.48	28.56

Table 3: Summary of Overhead Time (s)

SO-SA	DYCORS	LMSRBF	DDS	SSKm	ESGRBF	NOMADm	EGO
1.2×10^3	350	300	0.18	7.1×10^5	2.0	67	1.8×10^4

6. Conclusions

Response surface based global optimization algorithms have been playing a very important role for computationally expensive objective functions, arising from many practical problems, for example, parameter calibration of complex physical models. In this paper, a stochastic response surface method named “MSRS” proposed by Regis and Shoemaker (2007) where the next function evaluation point is chosen from a set of random candidate points, was further analyzed and extended. Specifically, we propose a new way to balance the local searching and global exploration by extending the way of generalizing the random candidate points via setting different probability values for each coordinate to be perturbed. The smaller probability value is more likely to prefer local searching and vice versa. Correspondingly we present a new definition of “neighborhood” by the number of perturbed coordinates. Based on the above new ideas, we finally proposed a specific implementation of “MSRS”, which takes sensitivity information into consideration when selecting the perturbed coordinate for producing the random candidate points. Its outstanding performance over many state of the art algorithms is well demonstrated by many typical testing problems, especially for high dimensional problems. In the future, we will further study the features of the method of the random candidate points and extend it to many other existing algorithms, for example, EGO, which does not work well for high dimensional problems in our experiments.

7. Acknowledgement

This work was supported by the Natural Science Foundation of China, Grant Nos. 11201054, 91330201 and by the Fundamental Research Funds for the Central Universities ZYGX2012J118, ZYGX2013Z005.

References

- Aleman, Dionne M., H. Edwin Romeijn, James F. Dempsey. 2009. A Response Surface Approach to Beam Orientation Optimization in Intensity-Modulated Radiation Therapy Treatment Planning. *INFORMS JOURNAL ON COMPUTING* **21** 62–76. doi:10.1287/ijoc.1080.0279. URL <http://joc.journal.informs.org/cgi/content/abstract/21/1/62>.
- ARNOLD, J. G., MORIASI D. N., GASSMAN P. W., ABBASPOUR K. C., WHITE M. J., SRINIVASAN R., SANTHI C., HARMEL R. D., GRIENSVEN A. Van, VAN LIEW M. W., KANNAN N., JHA M. K. 2012. Swat: Model use, calibration, and validation. *Transactions of the ASABE (Print)* **55** 1491–1508. URL <http://www.refdoc.fr/Detailnotice?idarticle=54492256>. Eng.
- Audet, Charles, J. E. Dennis, Jr. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. on Optimization* **17** 188–217. doi:10.1137/040603371. URL <http://portal.acm.org/citation.cfm?id=1132370.1132431>.
- Bäck, Thomas. 1995. Evolution strategies: an alternative evolutionary algorithm. *Artificial Evolution*. Springer-Verlag, 3–20.
- Björkman, M., K. Holmström. 2001. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering* **1** 373–397.
- Buhmann, M. D. 2003. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press.
- Conn, A., K. Scheinberg, Ph. Toint. 1997. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming* **79** 397–414. URL <http://dx.doi.org/10.1007/BF02614326>. 10.1007/BF02614326.
- Cortes, Corinna, Vladimir Vapnik. 1995. Support vector machine. *Machine learning* **20** 273–297.
- Cressie, N. 1991. *Statistics for spatial data*. Wiley-Interscience, New York.
- Dennis, J. E., Jr., Virginia Torczon. 1991. Direct search methods on parallel machines. *SIAM Journal on Optimization* **1** 448–474.
- Dennis, J. E., V. Torczon. 1995. *Managing approximation models in optimization*, chap. Multidisciplinary Design Optimization: State of the Art. SIAM, 330–347.
- Egea, José, Emmanuel Vazquez, Julio Banga, Rafael Marti. 2009. Improved scatter search for the global optimization of computationally expensive dynamic models. *Journal of Global Optimization* **43** 175–190. URL <http://hal.archives-ouvertes.fr/hal-00446227/en/>.
- Gilmore, P., C. T. Kelley. 1995. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM J. Optim* **5** 269–285.

- Glover, Fred. 1998. A template for scatter search and path relinking. *AE '97: Selected Papers from the Third European Conference on Artificial Evolution*. Springer-Verlag, London, UK, 3–54.
- Gutmann, H.-M. 2001. A radial basis function method for global optimization. *J. of Global Optimization* **19** 201–227. doi:<http://dx.doi.org/10.1023/A:1011255519438>.
- Hansen, Nikolaus, Sibylle D. Müller, Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evol. Comput.* **11** 1–18. doi:<http://dx.doi.org/10.1162/106365603321828970>. URL <http://dx.doi.org/10.1162/106365603321828970>.
- Holmström, Kenneth. 2008. An adaptive radial basis algorithm (arbf) for expensive black-box global optimization. *J. of Global Optimization* **41** 447–464. doi:<http://dx.doi.org/10.1007/s10898-007-9256-8>.
- Huyer, Waltraud, Arnold Neumaier. 1999. Global optimization by multi-level coordinate search. *Journal of Global Optimization* **14** 331–355. URL <http://dx.doi.org/10.1023/A:1008382309369>. 10.1023/A:1008382309369.
- Jones, D. R., C. D. Perttunen, B. E. Stuckman. 1993. Lipschitzian optimization without the lipschitz constant. *J. Optim. Theory Appl.* **79** 157–181. doi:10.1007/BF00941892. URL <http://portal.acm.org/citation.cfm?id=173669.173678>.
- Jones, D. R., M. Schonlau, W. J. Welch. 1998. Efficient global optimization of expensive black-box functions. *J. of Global Optimization* **13** 455–492. doi:<http://dx.doi.org/10.1023/A:1008306431147>.
- Jones, D.R. 2001. The direct global optimization algorithm. *Encyclopedia of Optimization, Kluwer Academic Publishers*.
- Kolda, Tamara G., Robert Michael Lewis, Virginia Torczon. 2003. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review* **45** 385–482.
- Laguna, M., R. Marti. 2003. *Scatter Search: Methodology and Implementations in C*. Kluwer Academic Publishers.
- Lophaven. S.N., H.B., Nielsen, J. Sondergaard. 2002. Dace: A matlab kriging toolbox, version 2.0. information and mathematical modeling. *Technical University of Denmark*.
- Pardalos, Panos M., Xin Liu. 1998. A tabu based pattern search method for the distance geometry problem. *New trends in mathematical programming*. Kluwer Acad. Publ., Boston, MA, 223–234.
- Powell, M.J.D. 1992. *The theory of radial basis function approximation in 1990*. Advances in Numerical Analysis II: Wavelets, Subdivision Algorithms, and Radial Basis Functions, Oxford University Press, Oxford.

- Powell, M.J.D. 2002. Uobyqa: Unconstrained optimization by quadratic approximation. *Mathematical Programming* **92** 555–582.
- Price, K., R. M. Storn, J. A. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*. Springer.
- Regis, R. G., C. A. Shoemaker. 2007. A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions. *INFORMS Journal on Computing* **19** 497–509. doi:10.1287/ijoc.1060.0182.
- Regis, R. G., C.A. Shoemaker. 2004. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation* **8** 490–505.
- Regis, R.G., C.A. Shoemaker. 2013. Dynamic coordinate search for expensive black-box optimization using radial basis functions. *Engineering Optimization* **45** 529–555. Submitted.
- Rinnooy Kan, A., G. Timmer. 1987. Stochastic global optimization methods part ii: Multi level methods. *Mathematical Programming* **39** 57–78. URL <http://dx.doi.org/10.1007/BF02592071>. 10.1007/BF02592071.
- Rios, L. M., N. V. Sahinidis. 2009. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Advanced in Optimization II*. AIChE.
- Sacks, J., W. J. Welch, T. J. Mitchell, H. P. Wynn. 1989. Design and analysis of computer experiments. *Statistical Science* **4** 409–423. doi:10.1214/ss/1177012413. URL <http://dx.doi.org/10.1214/ss/1177012413>.
- Shan, Songqing, G.Gary Wang. 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization* **41** 219–241. doi:10.1007/s00158-009-0420-2. URL <http://dx.doi.org/10.1007/s00158-009-0420-2>.
- Tolson, Bryan A., Christine A. Shoemaker. 2007a. Cannonsville reservoir watershed {SWAT2000} model development, calibration and validation. *Journal of Hydrology* **337** 68 – 86. doi:http://dx.doi.org/10.1016/j.jhydrol.2007.01.017. URL <http://www.sciencedirect.com/science/article/pii/S0022169407000273>.
- Tolson, Bryan A., Christine A. Shoemaker. 2007b. Dynamically dimensioned search algorithm for computationally efficient watershed model calibration. *Water Resour. Res.* **43**.
- Torczon, Virginia. 1997. On the convergence of pattern search algorithms. *SIAM J. on Optimization* **7** 1–25. doi:http://dx.doi.org/10.1137/S1052623493250780.
- Ugray, A., L Lasdon, J. Plummer, F. Glover, J. Kelly, R. Martí. 2007. Scatter search and local nlp solvers: A multistart framework for global optimization. *INFORMS Journal on Computing* **19** 328–340.
- Viana, F.A.C. 2010. *SURROGATES Toolbox User's Guide*. Gainesville, FL, USA, version 2.1 ed. URL <http://sites.google.com/site/felipeacviana/surrogatestoolbox>.

- Villemonteix, Julien, Emmanuel Vazquez, Eric Walter. 2009. An informational approach to the global optimization of expensive-to-evaluate functions. *J. of Global Optimization* **44** 509–534. doi:10.1007/s10898-008-9354-2. URL <http://dx.doi.org/10.1007/s10898-008-9354-2>.
- Ye, K. Q., W. Li, A. Sudjianto. 2000. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of Statistical Planning and Inference* **90** 145 – 159. doi: DOI:10.1016/S0378-3758(00)00105-1.
- Ye, Y. 1987. Interior algorithms for linear, quadratic and linearly constrained non-linear programming. Phd thesis, Stanford University.
- Zabinsky, Z.B. 2003. *Stochastic Adaptive Search for Global Optimization, Nonconvex Optimization and Its Applications*, vol. 72. Springer.
- Zaslavski, Alexander, M. Powell. 2006. The newuoa software for unconstrained optimization without derivatives. Panos Pardalos, G. Pillo, M. Roma, eds., *Large-Scale Nonlinear Optimization, Nonconvex Optimization and Its Applications*, vol. 83. Springer US, 255–297.

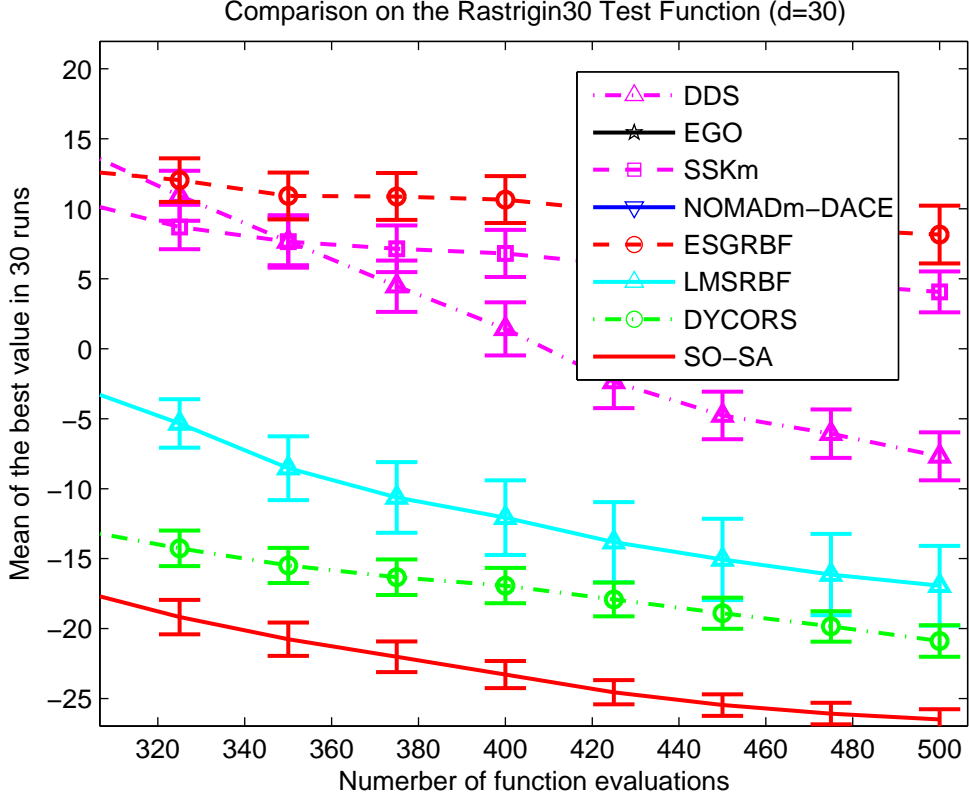
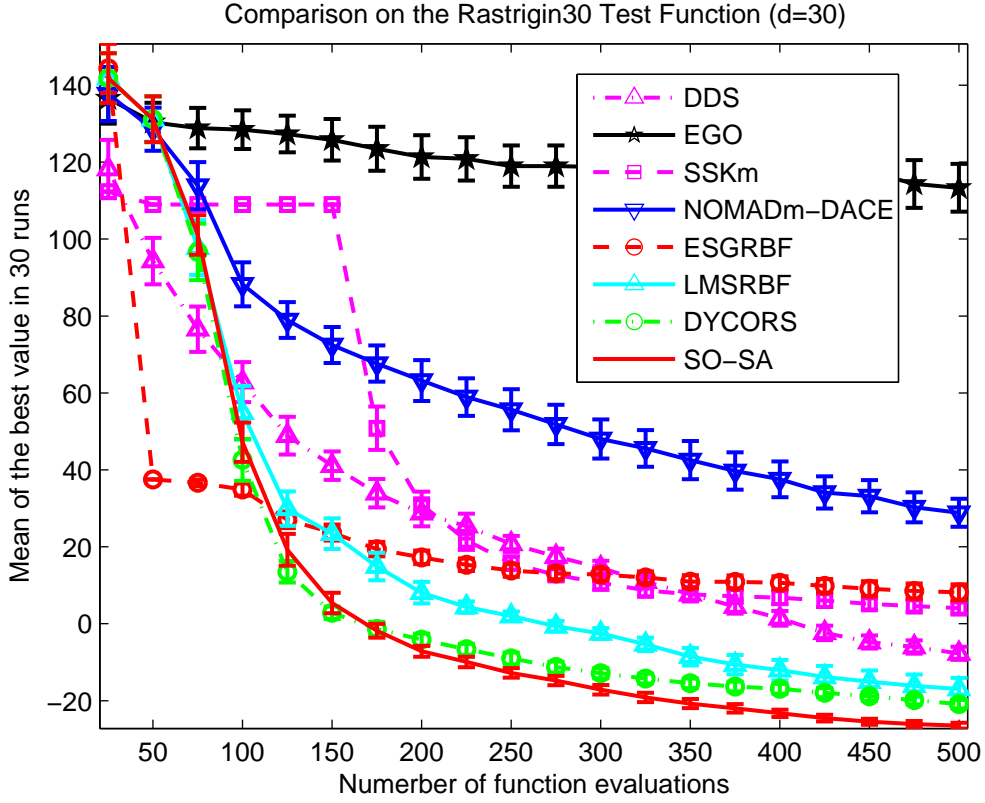


Figure 2: Comparison of Global Optimization Methods on the Rastrigin Function ($d = 30$)

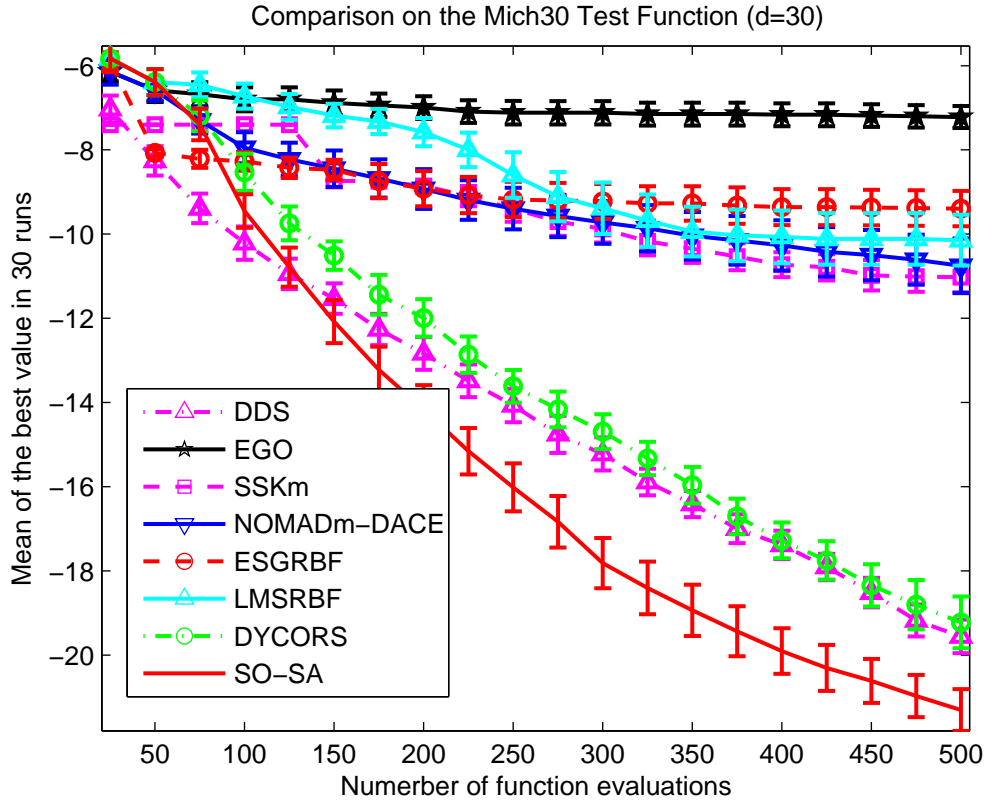


Figure 3: Comparison of Global Optimization Methods on the Michalewicz Function ($d = 30$)

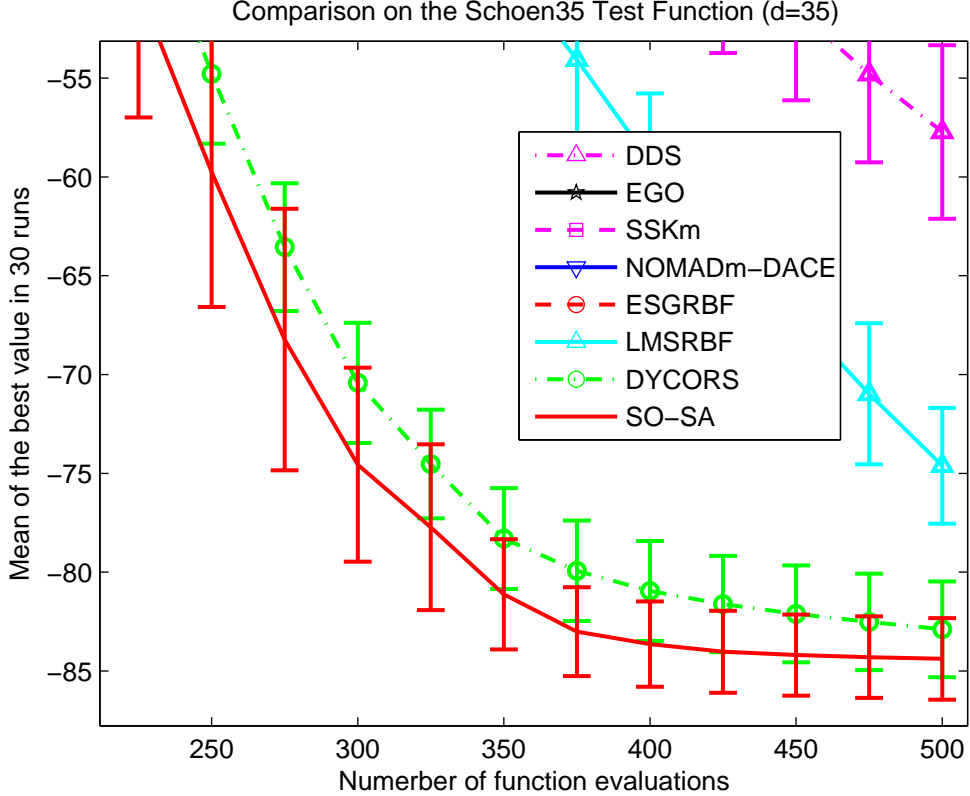
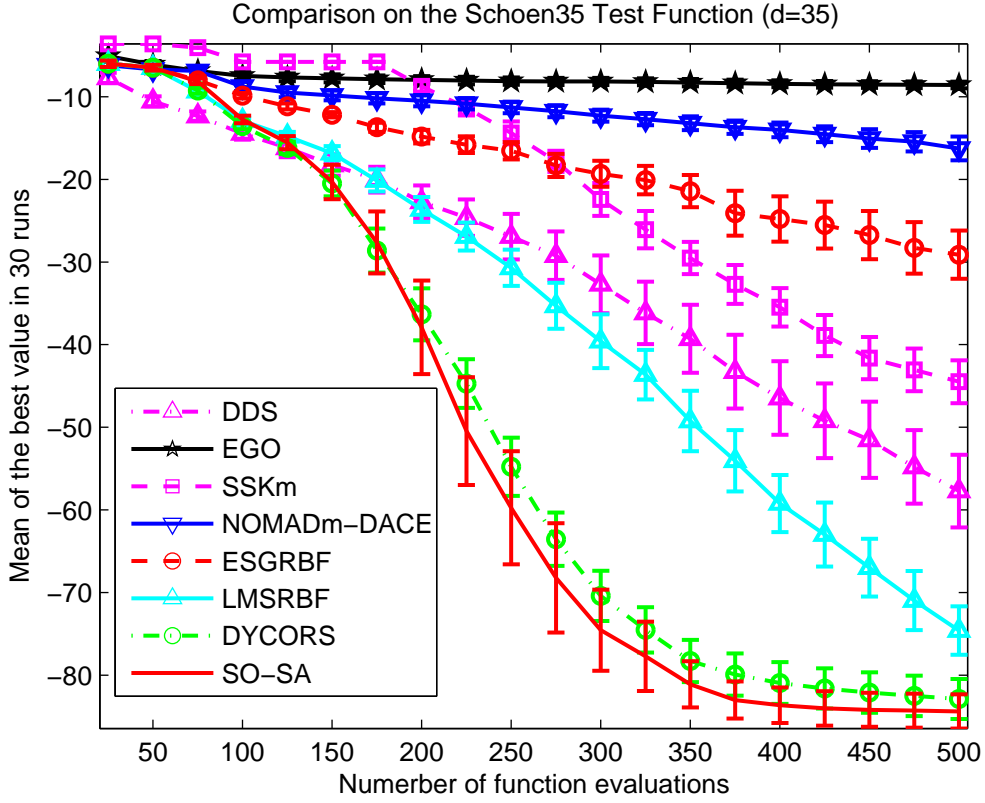


Figure 4: Comparison of Global Optimization Methods on the Schoen Function ($d = 35$). The lower one is the zoom in version.

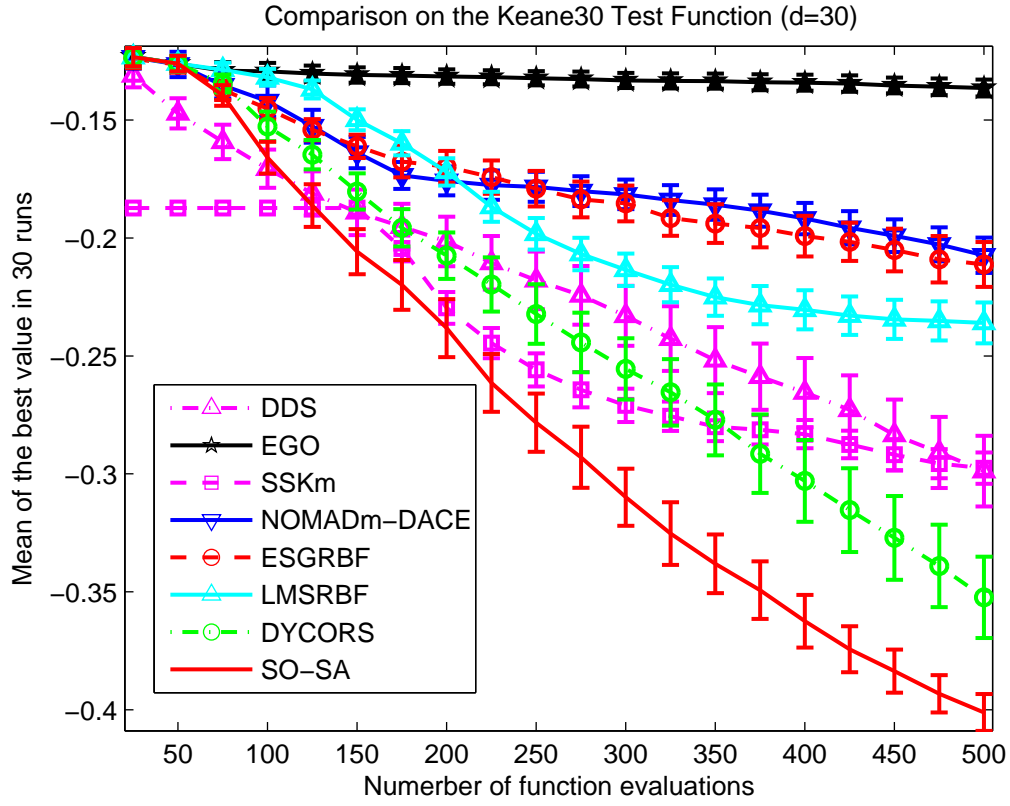


Figure 5: Comparison of Global Optimization Methods on the Keane Function ($d = 30$)

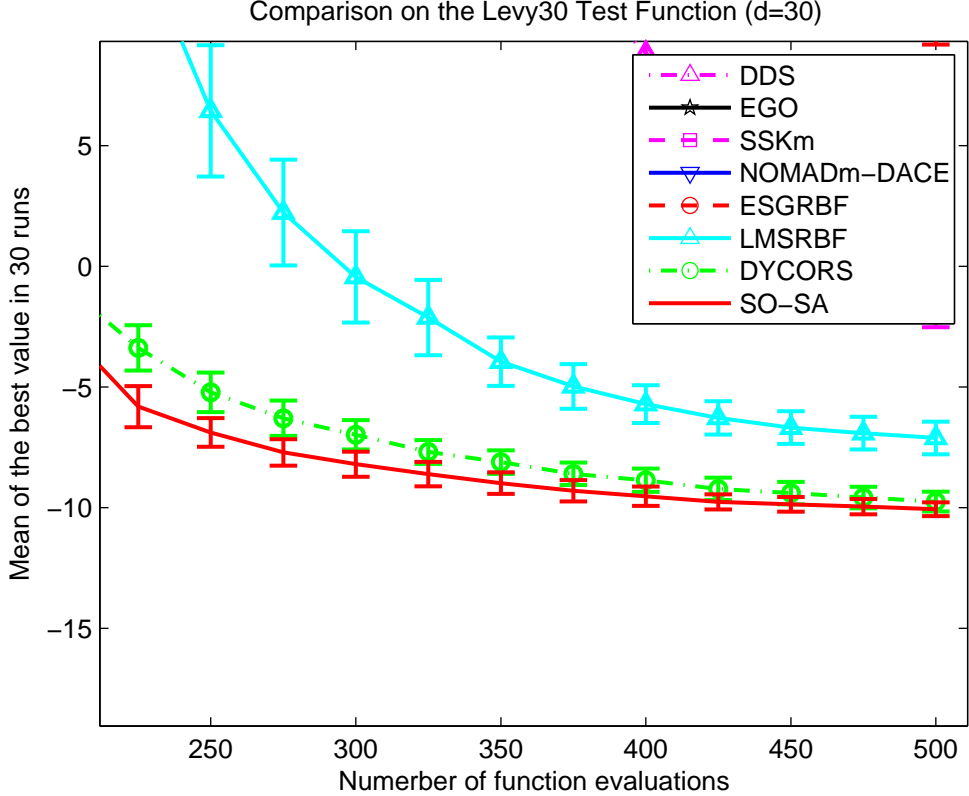
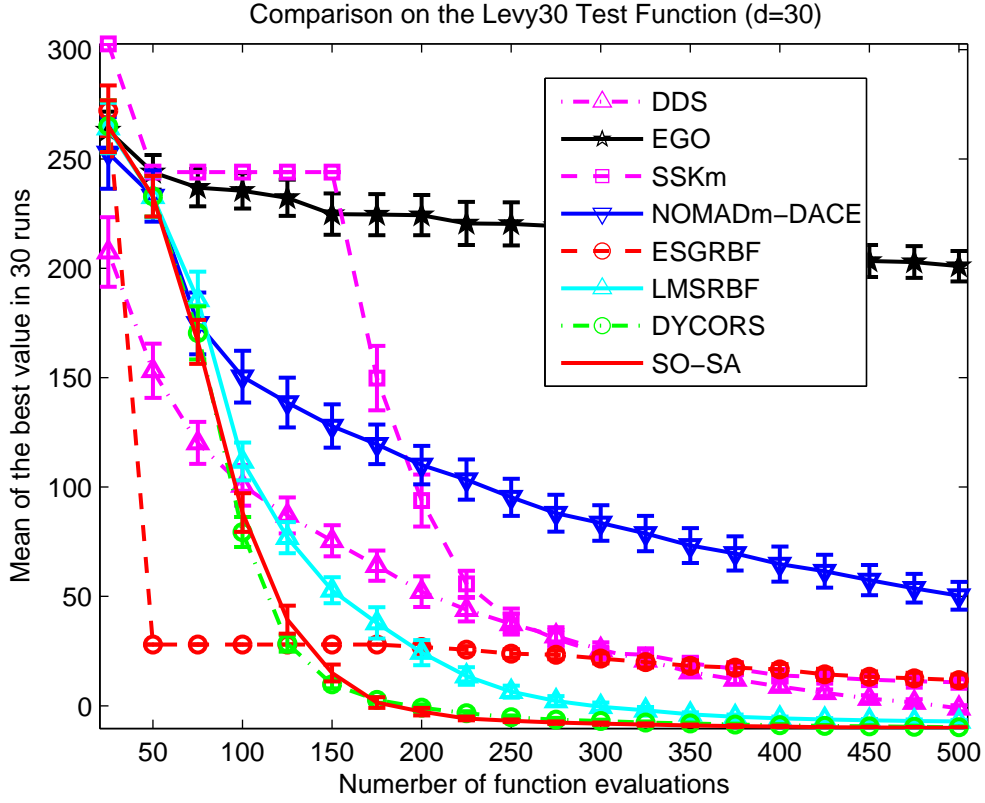


Figure 6: Comparison of Global Optimization Methods on the Levy Function ($d = 30$). The lower one is the zoom in version.

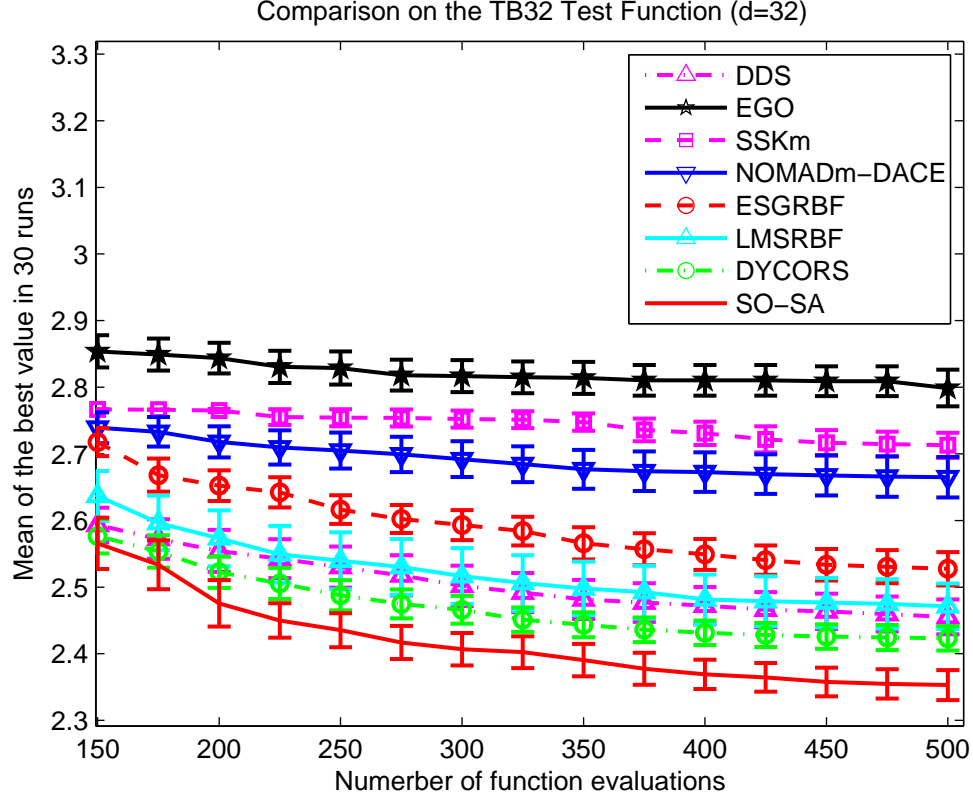
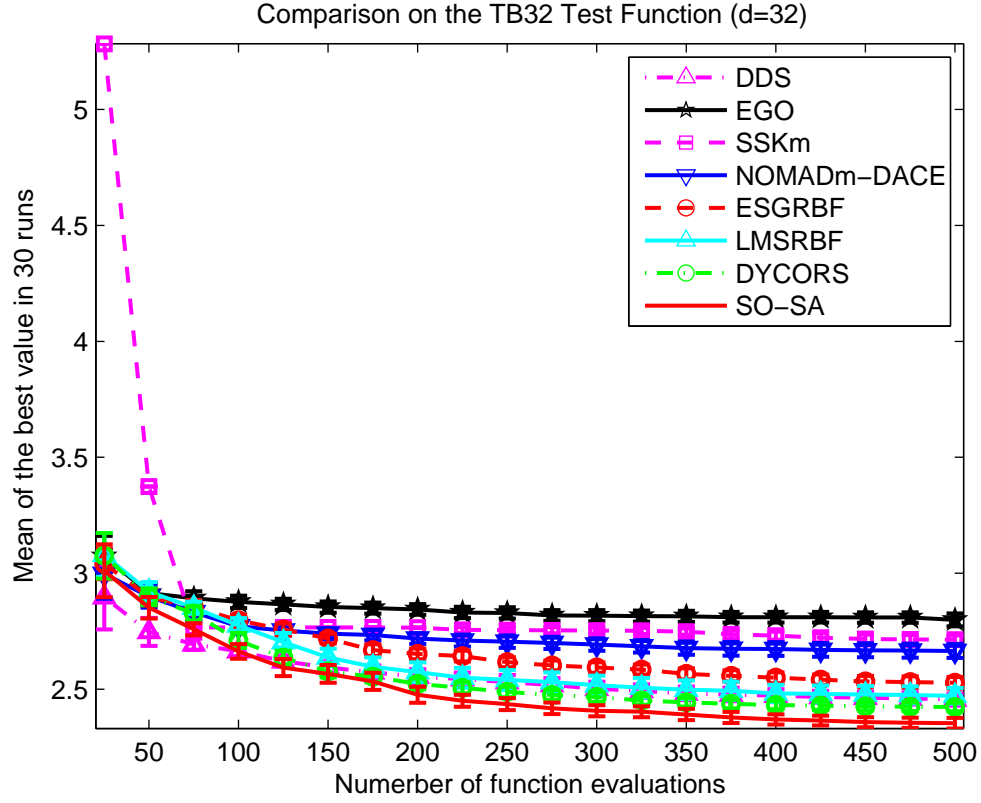
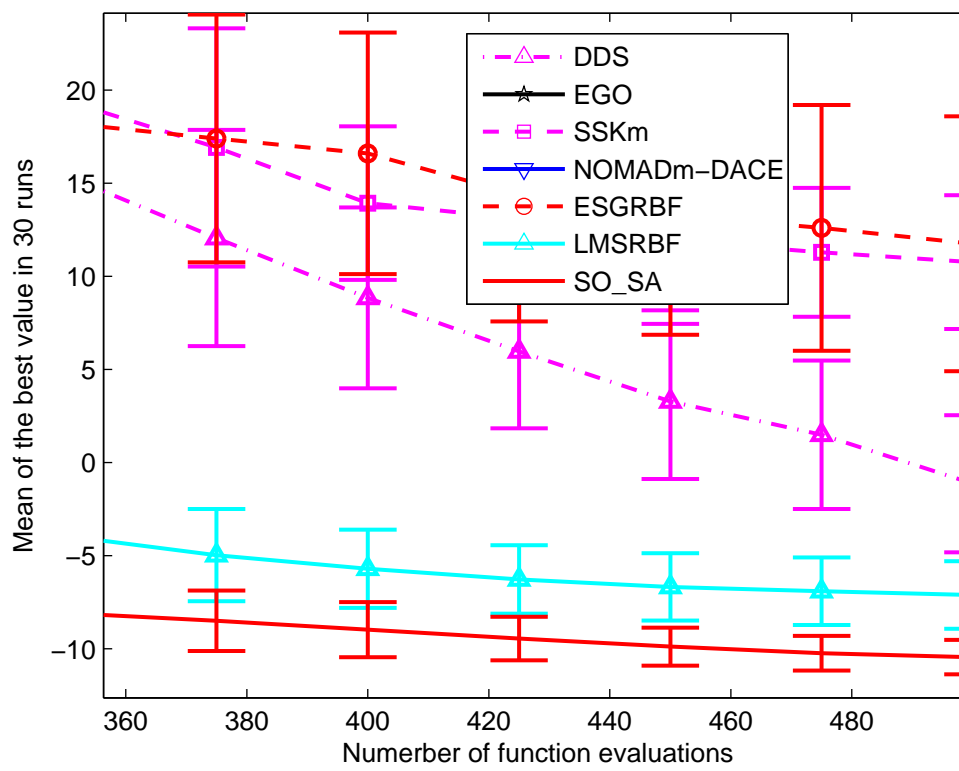


Figure 7: Comparison of Global Optimization Methods on the Town Brook Problem ($d = 32$). The lower one is the zoom in version.

Comparison on the Levy30 Test Function (d=30)



Comparison on the GWSS35₁₀₂ Test Function (d=35)

